ECE 344L Laboratory 3

Digital I/O

SPRING 2020

AUTHOR: DAVID KIRBY

DUE DATE: 05 MARCH 2020

Laboratory 3
Digital I/O and Finite State Machine

Due Date: 3 Mar 2020 Name: David Kirby

Points: 100 Points
Work individually.

Objective: The purposes of this laboratory are to develop your understanding of the PIC digital I/O capabilities, to begin familiarization of the Digilent **MX7** board I/O configuration, and to enhance your microcontroller programming skills, by implementing a finite state machine. In this lab you will begin using c function libraries that are included using the *#include <plib.h>* command in your c source file. The c library reference manuals are posted on UNM Learn.

Activities: For this assignment, you will write a **main** routine which implements a finite state machine. You will use the LEDs to indicate the state and output functions and the switches to provide the nickel and dime inputs. Your program will implement the functionality that is necessary to purchase an item which costs 30 cents.

You will assign state numbers to the individual states and light LED1, LED2, and LED3 to indicate the state number. (I recommend 0,1,2,3,..) Your initial state number will be 000, or all three LEDs unlit. You will use LED4 to indicate when a product has been dispensed. If change is to be issued, your program should flash LED1, LED2, and LED3 momentarily before they all return to the unlit condition. Your code should use the following state definitions:

State	Binary Representation
No money in - Start	000
5 cents inserted	001
10 cents inserted	010
15 cents inserted	011
20 cents inserted	100
25 cents inserted	101

You will simulate a nickel input using switch BTN1 (RG6) and a dime input using BTN2 (RG7). You will find that you need to use the delay function to ensure that you can observe the illumination of the LEDs and to correctly read the switches, as we will not be using interrupts to detect a change of input state. The delay will also prevent the system from cycling through the states if a button is held down too long.

You will use BTN3 as a reset button which will refund all money entered and reset the state to 0 if the system is not already in the zero state. When the reset is pressed, your program

Introduction

The objective of this lab was to introduce us to using the PIC32 Digital I/O, to introduce the Digilent MX7 board, and to implement a finite state machine using the C programming language.

Solution Methodology

For this assignment, we wrote a main routine that implemented a finite state machine. We used the LEDs of the MX7 to indicate the state and output functions and used the MX7 buttons to provide the nickel and dime inputs. Our program implemented the functionality that is necessary to purchase an item that costs 30 cents. We assigned state numbers to the individual states and lit LED1, LED2, and LED3 to indicate the state number. Our initial state number is 000, or all three LEDs unlit. We used LED4 to indicate when a product was dispensed. If change was issued, our program flashed LED1, LED2, and LED3 momentarily before they all returned to the unlit condition.

We simulated a nickel input using switch BTN1 (RG6) and a dime input using BTN2 (RG7). We needed to use the delay function to ensure we could observe the illumination of the LEDs and to correctly read the switches. We used BTN3 as a reset button which refunded all money entered and reset the state to 0. When the reset was pressed, our program flashed LED1, LED2, and LED3 momentarily before they all returned to the unlit condition. Figure 1 shows the state machine logic used to implement the program.

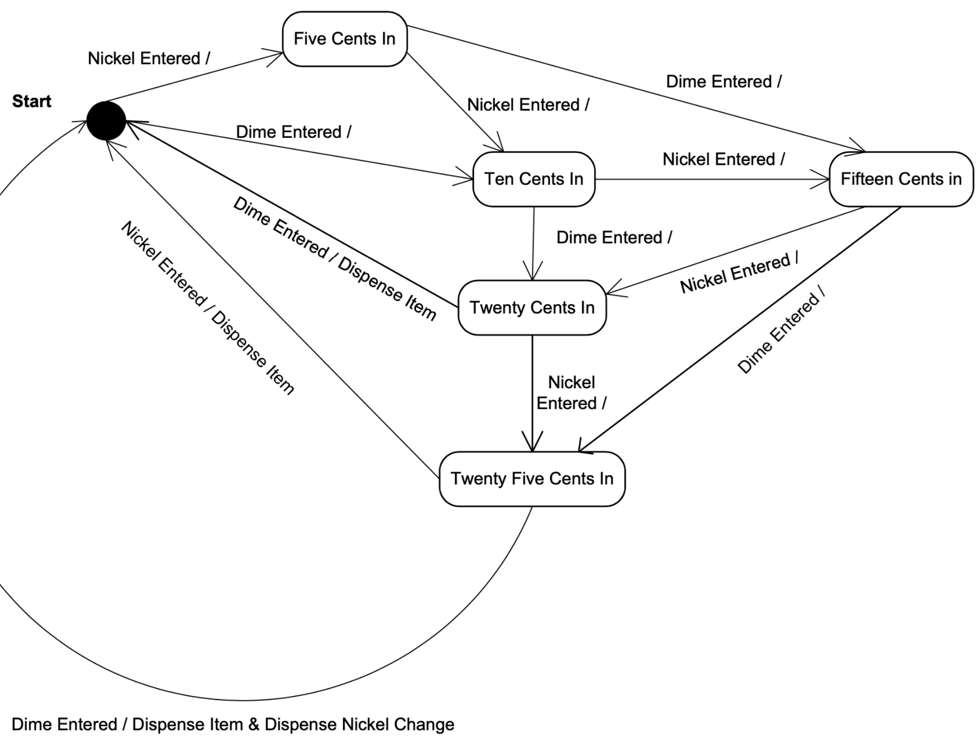

FIGURE 1

Source Code: kirby_lab03.c

```c
/**************************************************************************/
/*                                                                        */
/*      ECE 344L    -    Microprocessors    -    Spring 2020              */
/*                                                                        */
/*      kirby_lab03.c  -  Digital I/O and Finite State Machine            */
/*                                                                        */
/**************************************************************************/
/*                                                                        */
/*      Author: David Kirby                                               */
/*                                                                        */
/**************************************************************************/
/*                                                                        */
/*      File Description:                                                 */
/*          Implements a finite state machine using the LEDs and buttons on  */
/*          the chipKIT MX7 board.                                        */
/*                                                                        */
/**************************************************************************/
/*                                                                        */
/*      Revision History:                                                 */
/*          Original Source Code by: E.J. Nava, 9/23/18                   */
/*          Modified Code by: David Kirby, 01-Mar-2020                    */
/*                                                                        */
/**************************************************************************/

#include <plib.h>

/* ---------------------------------------------------------------------- */
/*                            Configuration Bits                          */
/* ---------------------------------------------------------------------- */

// Configure MX7 board for debugging
#pragma config ICESEL = ICS_PGx1

// SYSCLK = 80 MHz (8 MHz Crystal/ FPLLIDIV * FPLLMUL / FPLLODIV)
// Primary Osc w/PLL (XT+,HS+,EC+PLL)

#pragma config FPLLMUL = MUL_20, FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1
#pragma config POSCMOD = EC, FNOSC = PRIPLL, FPBDIV = DIV_8
#pragma config FSOSCEN = OFF            // Secondary oscillator enable
#define SYS_FREQ (80000000L)

// *** these are preconfigured on the MX4 Board for a clock frequency of 80MHz
// *** and a PBCLK value of 10MHz.

/* ---------------------------------------------------------------------- */
/*                            Forward Declarations                        */
/* ---------------------------------------------------------------------- */

void DeviceInit();
void DelayInit();
void DelayMs(int cms);
```

```c
void DisplayInit(int coins);


/* -------------------------------------------------------------------------- */
/*                                 Definitions                                */
/* -------------------------------------------------------------------------- */

#define cntMsDelay  10000           //timer 1 delay for 1ms

/* -------------------------------------------------------------------------- */
/*                                    Main                                    */
/* -------------------------------------------------------------------------- */

int main()
{
        int button_in12 = 0;
        int button_in3 = 0;
        int coins = 0;
        int msdelay = 100;

        //Set LD1 through LD4 as digital output
        DeviceInit();
        //Initialize timer for delay
        DelayInit();

        /* Perform the main application loop*/
        while (1)
        {
                // Read buttons
                button_in12 = PORTReadBits (IOPORT_G, BIT_6|BIT_7);
                button_in3  = PORTReadBits (IOPORT_A, BIT_0);

                if (button_in12 != 0)
                {
                        // drive both LD1 and LD2 high if both buttons pressed
                        if (((button_in12 & 0x0040) != 0) &&
                            ((button_in12 & 0x0080) != 0))
                                coins = coins+15;
                        else
                        {
                                //drive LD1 high if only BTN1 pressed
                                if ((button_in12 & 0x0040) !=0) // BTN1 pressed?
                                        coins = coins+5;
                                //drive LD2 high if only BTN2 pressed
                                if ((button_in12 & 0x0080) != 0) // BTN2 pressed
                                        coins = coins+10;
                        }
                }
                // Handle BTN3 separately
                if(button_in3 !=0)
                {
                    coins=0;
                    PORTWrite(IOPORT_G,BIT_12|BIT_13|BIT_14);
                    DelayMs(msdelay);
```

```
                    PORTClearBits(IOPORT_G,BIT_12|BIT_13| BIT_14|BIT_15);
                }
                //Initialize display
                DisplayInit(coins);

        }

}

/* ------------------------------------------------------------------------ */
/*    DisplayInit()
**
**    Parameters:
**        coins            -amount of money entered
**        delay            -delay between blinks
**
**    Return Value:
**        none
**
**    Errors:
**        none
**
**    Description:
**        Set display state based on amount of money entered
/* ------------------------------------------------------------------------ */

void DisplayInit(int coins)
{
        int msdelay = 230;
        int timeout=0;

        switch (coins)
        {
        case  5:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_12);   //001
                break;
        case 10:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_13);   //010
                break;
        case 15:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_12|BIT_13);   //011
                break;
        case 20:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
```

```
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_14);                           //100
                break;
        case 25:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                DelayMs(msdelay);
                PORTWrite (IOPORT_G, BIT_12|BIT_14);                    //101
                break;
        case 30:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                while(timeout<3)
                {
                      DelayMs(msdelay);
                      PORTWrite (IOPORT_G, BIT_15);                     //111
                      DelayMs(msdelay);
                      PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                      timeout++;
                }
                main();
                break;
        case 35:
                //DelayMs(msdelay);
                //PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                while(timeout<3)
                {
                        DelayMs(msdelay);
                        PORTWrite (IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);//111+
                        DelayMs(msdelay);
                        PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                        timeout++;
                }
                main();
                break;
        default:
                PORTClearBits(IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
                //Debug LEDs - send them all high at first, then trigger
                //PORTWrite (IOPORT_G, BIT_12|BIT_13|BIT_14|BIT_15);
        }
}

/* ------------------------------------------------------------------------ */
/*    DeviceInit()
**
**    Parameters:
**        none
**
**    Return Value:
**        none
**
**    Errors:
**        none
**
```

```
**    Description:
**        Set LD1 through LD4 as digital output
/* ------------------------------------------------------------------------ */

void DeviceInit()
{
        // On MX7 board, disable JTAG function
        DDPCONbits.JTAGEN = 0;

        //On MX7 LED1 is on RG12
        //       LED2 is on RG13
        //       LED3 is on RG14
        //       LED4 is on RG15
        //Set ports for onboard LEDs to outputs & clear them
        PORTSetPinsDigitalOut (IOPORT_G, BIT_12|BIT_13| BIT_14|BIT_15);
        PORTClearBits(IOPORT_G, BIT_12|BIT_13| BIT_14|BIT_15);
        //Set ports for onboard BTNs as inputs
        PORTSetPinsDigitalIn (IOPORT_G, BIT_6 | BIT_7);
        PORTSetPinsDigitalIn (IOPORT_A, BIT_0);
}

/* ------------------------------------------------------------------------ */
/*    DelayInit
**
**    Parameters:
**        none
**
**    Return Value:
**        none
**
**    Errors:
**        none
**
**    Description:
**        Initialized the hardware for use by delay functions. This
**        initializes Timer 1 to count at 10Mhz.
/* ------------------------------------------------------------------------ */

void DelayInit()
{
        unsigned int tcfg;

        /* Configure Timer 1 to count a 10MHz with a period of 0xFFFF*/
        tcfg = T1_ON|T1_IDLE_CON|T1_SOURCE_INT|T1_PS_1_1|T1_GATE_OFF|T1_SYNC_EXT_OFF;
        OpenTimer1(tcfg, 0xFFFF);
}

/* ------------------------------------------------------------------------ */
/*    DelayMs
**
**    Parameters:
**        cms            - number of milliseconds to delay
**
**    Return Value:
```

```
**          none
**
**      Errors:
**          none
**
**      Description:
**          Delay the requested number of milliseconds. Uses Timer1.
/* ------------------------------------------------------------------------------ */

void DelayMs(int cms)
{
        int ims;

        for (ims=0; ims<cms; ims++)
        {
                WriteTimer1(0);     // reset timer
                while (ReadTimer1() < cntMsDelay); // wait for interval of 1 mS
        }

}
```

Conclusion

Laboratory 3 was designed to introduce us to using the PIC32 Digital I/O, to introduce the Digilent MX7 board, and to implement a finite state machine using the C programming language. This gave us more practice with MPLab by learning to implement state machines and Digital I/O. This was critical to understanding how to properly retrieve feedback as necessary and challenged us to implement delays in order to get an accurate responses.